

| | | | | | | | |
|----------|----|------|--------|------------|----|------------|----------|
| RRRRRRRR | MM | MM | 222222 | PPPPPPPP | UU | UU | TTTTTTTT |
| RRRRRRRR | MM | MM | 222222 | PPPPPPPP | UU | UU | TTTTTTTT |
| RR RR | RR | MMMM | MMMM | 22 | PP | PP | UU |
| RR RR | RR | MMMM | MMMM | 22 | PP | PP | UU |
| RR RR | RR | MM | MM | 22 | PP | PP | UU |
| RR RR | RR | MM | MM | 22 | PP | PP | UU |
| RRRRRRRR | MM | MM | 22 | PPPPPPPP | UU | UU | TT |
| RRRRRRRR | MM | MM | 22 | PPPPPPPP | UU | UU | TT |
| RR RR | RR | MM | MM | 22 | PP | UU | UU |
| RR RR | RR | MM | MM | 22 | PP | UU | UU |
| RR RR | RR | MM | MM | 22 | PP | UU | UU |
| RR RR | RR | MM | MM | 22 | PP | UU | UU |
| RR RR | RR | MM | MM | 2222222222 | PP | UUUUUUUUUU | TT |
| RR RR | RR | MM | MM | 2222222222 | PP | UUUUUUUUUU | TT |

| | | |
|----------|--|---------|
| LL | | SSSSSSS |
| LL | | SSSSSSS |
| LL | | SS |
| LL | | SS |
| LL | | SS |
| LL | | SSSSS |
| LL | | SSSSS |
| LL | | SS |
| LL | | SS |
| LL | | SS |
| LLLLLLLL | | SSSSSSS |
| LLLLLLLL | | SSSSSSS |

(2) 145
(3) 178
(5) 448

DECLARATIONS
RMSPUT2 - HIGH LEVEL RELATIVE \$PUT
RMSPUTUPD2 - COMMON \$PUT AND \$UPDATE RELATIVE ROUTINE

1 \$BEGIN RM2PUT,000,RMSRMS2,<RELATIVE SPECIFIC PUT>
2
3 :*****
4 :
5 : COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
6 : DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
7 : ALL RIGHTS RESERVED.
8 :
9 :
10 : THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 : ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 : INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 : COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 : OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 : TRANSFERRED.
16 :
17 : THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 : AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 : CORPORATION.
20 :
21 : DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 : SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 :
24 :
25 :*****
26 :
27 :++
28 : Facility: RMS32
29 :
30 : Abstract:
31 : This module provides relative file organization
32 : specific processing for the \$PUT function.
33 :
34 :
35 : Environment:
36 : Star processor running Starlet exec.
37 :
38 : Author: L. F. Laverdure Creation Date: 7-NOV-1977
39 :
40 : Modified By:
41 :
42 : V03-019 JEJ0042 J E Johnson 21-Jun-1984
43 : Correct error in record locking code path that
44 : attempts to deallocate BDB twice if WAT bit is set
45 : and a locking error occurs.
46 :
47 : V03-018 DGB0016 Donald G. Blair 02-Mar-1984
48 : Allocate full-length FIB to support access mode
49 : protected files.
50 :
51 : V03-017 DAS0001 David Solomon 25-Jul-1983
52 : Fix a bug introduced in V03-016 that broke BI journaling
53 : of puts and updates.
54 :
55 : V03-016 KPL0005 Peter Lieberwirth 20-Jun-1983
56 : Change some references to JNLFLG to JNLFLG2.
57 :
58 :
59 :
60 :
61 :
62 :
63 :
64 :
65 :
66 :
67 :
68 :
69 :
70 :
71 :
72 :
73 :
74 :
75 :
76 :
77 :
78 :
79 :
80 :
81 :
82 :
83 :
84 :
85 :
86 :
87 :
88 :
89 :
90 :
91 :
92 :
93 :
94 :
95 :
96 :
97 :
98 :
99 :
100 :
101 :
102 :
103 :
104 :
105 :
106 :
107 :
108 :
109 :
110 :
111 :
112 :
113 :
114 :
115 :
116 :
117 :
118 :
119 :
120 :
121 :
122 :
123 :
124 :
125 :
126 :
127 :
128 :
129 :
130 :
131 :
132 :
133 :
134 :
135 :
136 :
137 :
138 :
139 :
140 :
141 :
142 :
143 :
144 :
145 :
146 :
147 :
148 :
149 :
150 :
151 :
152 :
153 :
154 :
155 :
156 :
157 :
158 :
159 :
160 :
161 :
162 :
163 :
164 :
165 :
166 :
167 :
168 :
169 :
170 :
171 :
172 :
173 :
174 :
175 :
176 :
177 :
178 :
179 :
180 :
181 :
182 :
183 :
184 :
185 :
186 :
187 :
188 :
189 :
190 :
191 :
192 :
193 :
194 :
195 :
196 :
197 :
198 :
199 :
200 :
201 :
202 :
203 :
204 :
205 :
206 :
207 :
208 :
209 :
210 :
211 :
212 :
213 :
214 :
215 :
216 :
217 :
218 :
219 :
220 :
221 :
222 :
223 :
224 :
225 :
226 :
227 :
228 :
229 :
230 :
231 :
232 :
233 :
234 :
235 :
236 :
237 :
238 :
239 :
240 :
241 :
242 :
243 :
244 :
245 :
246 :
247 :
248 :
249 :
250 :
251 :
252 :
253 :
254 :
255 :
256 :
257 :
258 :
259 :
260 :
261 :
262 :
263 :
264 :
265 :
266 :
267 :
268 :
269 :
270 :
271 :
272 :
273 :
274 :
275 :
276 :
277 :
278 :
279 :
280 :
281 :
282 :
283 :
284 :
285 :
286 :
287 :
288 :
289 :
290 :
291 :
292 :
293 :
294 :
295 :
296 :
297 :
298 :
299 :
300 :
301 :
302 :
303 :
304 :
305 :
306 :
307 :
308 :
309 :
310 :
311 :
312 :
313 :
314 :
315 :
316 :
317 :
318 :
319 :
320 :
321 :
322 :
323 :
324 :
325 :
326 :
327 :
328 :
329 :
330 :
331 :
332 :
333 :
334 :
335 :
336 :
337 :
338 :
339 :
340 :
341 :
342 :
343 :
344 :
345 :
346 :
347 :
348 :
349 :
350 :
351 :
352 :
353 :
354 :
355 :
356 :
357 :
358 :
359 :
360 :
361 :
362 :
363 :
364 :
365 :
366 :
367 :
368 :
369 :
370 :
371 :
372 :
373 :
374 :
375 :
376 :
377 :
378 :
379 :
380 :
381 :
382 :
383 :
384 :
385 :
386 :
387 :
388 :
389 :
390 :
391 :
392 :
393 :
394 :
395 :
396 :
397 :
398 :
399 :
400 :
401 :
402 :
403 :
404 :
405 :
406 :
407 :
408 :
409 :
410 :
411 :
412 :
413 :
414 :
415 :
416 :
417 :
418 :
419 :
420 :
421 :
422 :
423 :
424 :
425 :
426 :
427 :
428 :
429 :
430 :
431 :
432 :
433 :
434 :
435 :
436 :
437 :
438 :
439 :
440 :
441 :
442 :
443 :
444 :
445 :
446 :
447 :
448 :
449 :
450 :
451 :
452 :
453 :
454 :
455 :
456 :
457 :
458 :
459 :
460 :
461 :
462 :
463 :
464 :
465 :
466 :
467 :
468 :
469 :
470 :
471 :
472 :
473 :
474 :
475 :
476 :
477 :
478 :
479 :
480 :
481 :
482 :
483 :
484 :
485 :
486 :
487 :
488 :
489 :
490 :
491 :
492 :
493 :
494 :
495 :
496 :
497 :
498 :
499 :
500 :
501 :
502 :
503 :
504 :
505 :
506 :
507 :
508 :
509 :
510 :
511 :
512 :
513 :
514 :
515 :
516 :
517 :
518 :
519 :
520 :
521 :
522 :
523 :
524 :
525 :
526 :
527 :
528 :
529 :
530 :
531 :
532 :
533 :
534 :
535 :
536 :
537 :
538 :
539 :
540 :
541 :
542 :
543 :
544 :
545 :
546 :
547 :
548 :
549 :
550 :
551 :
552 :
553 :
554 :
555 :
556 :
557 :
558 :
559 :
560 :
561 :
562 :
563 :
564 :
565 :
566 :
567 :
568 :
569 :
570 :
571 :
572 :
573 :
574 :
575 :
576 :
577 :
578 :
579 :
580 :
581 :
582 :
583 :
584 :
585 :
586 :
587 :
588 :
589 :
590 :
591 :
592 :
593 :
594 :
595 :
596 :
597 :
598 :
599 :
600 :
601 :
602 :
603 :
604 :
605 :
606 :
607 :
608 :
609 :
610 :
611 :
612 :
613 :
614 :
615 :
616 :
617 :
618 :
619 :
620 :
621 :
622 :
623 :
624 :
625 :
626 :
627 :
628 :
629 :
630 :
631 :
632 :
633 :
634 :
635 :
636 :
637 :
638 :
639 :
640 :
641 :
642 :
643 :
644 :
645 :
646 :
647 :
648 :
649 :
650 :
651 :
652 :
653 :
654 :
655 :
656 :
657 :
658 :
659 :
660 :
661 :
662 :
663 :
664 :
665 :
666 :
667 :
668 :
669 :
670 :
671 :
672 :
673 :
674 :
675 :
676 :
677 :
678 :
679 :
680 :
681 :
682 :
683 :
684 :
685 :
686 :
687 :
688 :
689 :
690 :
691 :
692 :
693 :
694 :
695 :
696 :
697 :
698 :
699 :
700 :
701 :
702 :
703 :
704 :
705 :
706 :
707 :
708 :
709 :
710 :
711 :
712 :
713 :
714 :
715 :
716 :
717 :
718 :
719 :
720 :
721 :
722 :
723 :
724 :
725 :
726 :
727 :
728 :
729 :
730 :
731 :
732 :
733 :
734 :
735 :
736 :
737 :
738 :
739 :
740 :
741 :
742 :
743 :
744 :
745 :
746 :
747 :
748 :
749 :
750 :
751 :
752 :
753 :
754 :
755 :
756 :
757 :
758 :
759 :
760 :
761 :
762 :
763 :
764 :
765 :
766 :
767 :
768 :
769 :
770 :
771 :
772 :
773 :
774 :
775 :
776 :
777 :
778 :
779 :
780 :
781 :
782 :
783 :
784 :
785 :
786 :
787 :
788 :
789 :
790 :
791 :
792 :
793 :
794 :
795 :
796 :
797 :
798 :
799 :
800 :
801 :
802 :
803 :
804 :
805 :
806 :
807 :
808 :
809 :
8010 :
8011 :
8012 :
8013 :
8014 :
8015 :
8016 :
8017 :
8018 :
8019 :
8020 :
8021 :
8022 :
8023 :
8024 :
8025 :
8026 :
8027 :
8028 :
8029 :
8030 :
8031 :
8032 :
8033 :
8034 :
8035 :
8036 :
8037 :
8038 :
8039 :
8040 :
8041 :
8042 :
8043 :
8044 :
8045 :
8046 :
8047 :
8048 :
8049 :
8050 :
8051 :
8052 :
8053 :
8054 :
8055 :
8056 :
8057 :
8058 :
8059 :
8060 :
8061 :
8062 :
8063 :
8064 :
8065 :
8066 :
8067 :
8068 :
8069 :
8070 :
8071 :
8072 :
8073 :
8074 :
8075 :
8076 :
8077 :
8078 :
8079 :
8080 :
8081 :
8082 :
8083 :
8084 :
8085 :
8086 :
8087 :
8088 :
8089 :
8090 :
8091 :
8092 :
8093 :
8094 :
8095 :
8096 :
8097 :
8098 :
8099 :
80100 :
80101 :
80102 :
80103 :
80104 :
80105 :
80106 :
80107 :
80108 :
80109 :
80110 :
80111 :
80112 :
80113 :
80114 :
80115 :
80116 :
80117 :
80118 :
80119 :
80120 :
80121 :
80122 :
80123 :
80124 :
80125 :
80126 :
80127 :
80128 :
80129 :
80130 :
80131 :
80132 :
80133 :
80134 :
80135 :
80136 :
80137 :
80138 :
80139 :
80140 :
80141 :
80142 :
80143 :
80144 :
80145 :
80146 :
80147 :
80148 :
80149 :
80150 :
80151 :
80152 :
80153 :
80154 :
80155 :
80156 :
80157 :
80158 :
80159 :
80160 :
80161 :
80162 :
80163 :
80164 :
80165 :
80166 :
80167 :
80168 :
80169 :
80170 :
80171 :
80172 :
80173 :
80174 :
80175 :
80176 :
80177 :
80178 :
80179 :
80180 :
80181 :
80182 :
80183 :
80184 :
80185 :
80186 :
80187 :
80188 :
80189 :
80190 :
80191 :
80192 :
80193 :
80194 :
80195 :
80196 :
80197 :
80198 :
80199 :
80200 :
80201 :
80202 :
80203 :
80204 :
80205 :
80206 :
80207 :
80208 :
80209 :
80210 :
80211 :
80212 :
80213 :
80214 :
80215 :
80216 :
80217 :
80218 :
80219 :
80220 :
80221 :
80222 :
80223 :
80224 :
80225 :
80226 :
80227 :
80228 :
80229 :
80230 :
80231 :
80232 :
80233 :
80234 :
80235 :
80236 :
80237 :
80238 :
80239 :
80240 :
80241 :
80242 :
80243 :
80244 :
80245 :
80246 :
80247 :
80248 :
80249 :
80250 :
80251 :
80252 :
80253 :
80254 :
80255 :
80256 :
80257 :
80258 :
80259 :
80260 :
80261 :
80262 :
80263 :
80264 :
80265 :
80266 :
80267 :
80268 :
80269 :
80270 :
80271 :
80272 :
80273 :
80274 :
80275 :
80276 :
80277 :
80278 :
80279 :
80280 :
80281 :
80282 :
80283 :
80284 :
80285 :
80286 :
80287 :
80288 :
80289 :
80290 :
80291 :
80292 :
80293 :
80294 :
80295 :
80296 :
80297 :
80298 :
80299 :
80300 :
80301 :
80302 :
80303 :
80304 :
80305 :
80306 :
80307 :
80308 :
80309 :
80310 :
80311 :
80312 :
80313 :
80314 :
80315 :
80316 :
80317 :
80318 :
80319 :
80320 :
80321 :
80322 :
80323 :
80324 :
80325 :
80326 :
80327 :
80328 :
80329 :
80330 :
80331 :
80332 :
80333 :
80334 :
80335 :
80336 :
80337 :
80338 :
80339 :
80340 :
80341 :
80342 :
80343 :
80344 :
80345 :
80346 :
80347 :
80348 :
80349 :
80350 :
80351 :
80352 :
80353 :
80354 :
80355 :
80356 :
80357 :
80358 :
80359 :
80360 :
80361 :
80362 :
80363 :
80364 :
80365 :
80366 :
80367 :
80368 :
80369 :
80370 :
80371 :
80372 :
80373 :
80374 :
80375 :
80376 :
80377 :
80378 :
80379 :
80380 :
80381 :
80382 :
80383 :
80384 :
80385 :
80386 :
80387 :
80388 :
80389 :
80390 :
80391 :
80392 :
80393 :
80394 :
80395 :
80396 :
80397 :
80398 :
80399 :
80400 :
80401 :
80402 :
80403 :
80404 :
80405 :
80406 :
80407 :
80408 :
80409 :
80410 :
80411 :
80412 :
80413 :
80414 :
80415 :
80416 :
80417 :
80418 :
80419 :
80420 :
80421 :
80422 :
80423 :
80424 :
80425 :
80426 :
80427 :
80428 :
80429 :
80430 :
80431 :
80432 :
80433 :
80434 :
80435 :
80436 :
80437 :
80438 :
80439 :
80440 :
80441 :
80442 :
80443 :
80444 :
80445 :
80446 :
80447 :
80448 :
80449 :
80450 :
80451 :
80452 :
80453 :
80454 :
80455 :
80456 :
80457 :
80458 :
80459 :
80460 :
80461 :
80462 :
80463 :
80464 :
80465 :
80466 :
80467 :
80468 :
80469 :
80470 :
80471 :
80472 :
80473 :
80474 :
80475 :
80476 :
80477 :
80478 :
80479 :
80480 :
80481 :
80482 :
80483 :
80484 :
80485 :
80486 :
80487 :
80488 :
80489 :
80490 :
80491 :
80492 :
80493 :
80494 :
80495 :
80496 :
80497 :
80498 :
80499 :
80500 :
80501 :
80502 :
80503 :
80504 :
80505 :
80506 :
80507 :
80508 :
80509 :
80510 :
80511 :
80512 :
80513 :
80514 :
80515 :
80516 :
80517 :
80518 :
80519 :
80520 :
80521 :
80522 :
80523 :
80524 :
80525 :
80526 :
80527 :
80528 :
80529 :
80530 :
80531 :
80532 :
80533 :
80534 :
80535 :
80536 :
80537 :
80538 :
80539 :
80540 :
80541 :
80542 :
80543 :
80544 :
80545 :
80546 :
80547 :
80548 :
80549 :
80550 :
80551 :
80552 :
80553 :
80554 :
80555 :
80556 :
80557 :
80558 :
80559 :
80560 :
80561 :
80562 :
80563 :
80564 :
80565 :
80566 :
80567 :
80568 :
80569 :
80570 :
80571 :
80572 :
80573 :
80574 :
80575 :
80576 :
80577 :
80578 :
80579 :
80580 :
80581 :
80582 :
80583 :
80584 :
80585 :
80586 :
80587 :
80588 :
80589 :
80590 :
80591 :
80592 :
80593 :
80594 :
80595 :
80596 :
80597 :
80598 :
80599 :
80600 :
80601 :
80602 :
80603 :
80604 :
80605 :
80606 :
80607 :
80608 :
80609 :
80610 :
80611 :
80612 :
80613 :
80614 :
80615 :
80616 :
80617 :
80618 :
80619 :
80620 :
80621 :
80622 :
80623 :
80624 :
80625 :
80626 :
80627 :
80628 :
80629 :
80630 :
80631 :
80632 :
80633 :
80634 :
80635 :
80636 :
80637 :
80638 :
80639 :
80640 :
80641 :
80642 :
80643 :
80644 :
80645 :
80646 :
80647 :
80648 :
80649 :
80650 :
80651 :
80652 :
80653 :
80654 :
80655 :
80656 :
80657 :
80658 :
80659 :
80660 :
80661 :
80662 :
80663 :
80664 :
80665 :
80666 :
80667 :
80668 :
80669 :
80670 :
80671 :
80672 :
80673 :
80674 :
80675 :
80676 :
80677 :
80678 :
80679 :
80680 :
80681 :
80682 :
80683 :
80684 :
80685 :
80686 :
80687 :
80688 :
80689 :
80690 :
80691 :
80692 :
80693 :
80694 :
80695 :
80696 :
80697 :
80698 :
80699 :
80700 :
80701 :
80702 :
80703 :
80704 :
80705 :
80706 :
80707 :
80708 :
80709 :
80710 :
80711 :
807

| | | | | | |
|------|-----|---------|----------|---|-------------|
| 0000 | 58 | V03-015 | KPL0004 | Peter Lieberwirth | 26-May-1983 |
| 0000 | 59 | | | Fix RJR references for new format. | |
| 0000 | 60 | | | | |
| 0000 | 61 | V03-014 | JWH0206 | Jeffrey W. Horn | 12-Apr-1983 |
| 0000 | 62 | | | Fix bug in JWH0192 which was causing a call to LOCK | |
| 0000 | 63 | | | instead of QUERY_LOCK when WAT was specified for a \$PUT. | |
| 0000 | 64 | | | | |
| 0000 | 65 | V03-013 | KPL0003 | Peter Lieberwirth | 30-Mar-1983 |
| 0000 | 66 | | | AP is used as a flag to RMSPUTUPD2 to indicate whether the | |
| 0000 | 67 | | | operation is a PUT or an UPDATE. Due to an unrelated change, | |
| 0000 | 68 | | | this flag was sometimes set incorrectly when the operation | |
| 0000 | 69 | | | is a PUT. Fix this by plugging the AP. | |
| 0000 | 70 | | | | |
| 0000 | 71 | V03-012 | RAS0135 | Ron Schaefer | 17-Mar-1983 |
| 0000 | 72 | | | Corrections to RAS0132 for registers and RJRS_ names. | |
| 0000 | 73 | | | | |
| 0000 | 74 | V03-011 | RAS0132 | Ron Schaefer | 16-Mar-1983 |
| 0000 | 75 | | | Merge \$RMSRDEF into \$RJRDEF and revise the interface | |
| 0000 | 76 | | | for RM\$WRTJNL for easier use from ISAM. | |
| 0000 | 77 | | | | |
| 0000 | 78 | V03-010 | JWH0192 | Jeffrey W. Horn | 28-Feb-1983 |
| 0000 | 79 | | | Fix bucheck in \$PUT with WAT option. | |
| 0000 | 80 | | | | |
| 0000 | 81 | V03-009 | SPR52290 | Jeffrey W. Horn | 03-Jan-1983 |
| 0000 | 82 | | | Fix bugcheck in auto-extend with shared files. | |
| 0000 | 83 | | | | |
| 0000 | 84 | V03-008 | KPL0002 | Peter Lieberwirth | 7-Nov-1982 |
| 0000 | 85 | | | Fix RMSR name again. | |
| 0000 | 86 | | | | |
| 0000 | 87 | V03-007 | JWH0121 | Jeffrey W. Horn | 04-Nov-1982 |
| 0000 | 88 | | | Fix bug in journal logic that was causing non-journal | |
| 0000 | 89 | | | \$PUT to return a zero status. | |
| 0000 | 90 | | | | |
| 0000 | 91 | V03-006 | KPL0001 | Peter Lieberwirth | 26-Oct-1982 |
| 0000 | 92 | | | Correct size of RJR overhead added to R3 for call to | |
| 0000 | 93 | | | WRTJNL. Change RMSR names. | |
| 0000 | 94 | | | | |
| 0000 | 95 | V03-005 | JWH0112 | Jeffrey W. Horn | 06-Oct-1982 |
| 0000 | 96 | | | Implement new RJR format. Put in code for RU | |
| 0000 | 97 | | | journal support. | |
| 0000 | 98 | | | | |
| 0000 | 99 | V03-004 | KBT0129 | Keith B. Thompson | 20-Aug-1982 |
| 0000 | 100 | | | Reorganize psects | |
| 0000 | 101 | | | | |
| 0000 | 102 | V03-003 | KBT0117 | Keith B. Thompson | 6-Aug-1982 |
| 0000 | 103 | | | Remove ref. to set_sifb_adr and correct jeff's revision numbers | |
| 0000 | 104 | | | | |
| 0000 | 105 | V03-002 | | | |
| 0000 | 106 | | | There was an un-audited change done by JWH some time | |
| 0000 | 107 | | | around here to fix a bug introduced by JWH0001. | |
| 0000 | 108 | V03-001 | JWH0001 | Jeffrey W. Horn | 18-May-1982 |
| 0000 | 109 | | | Put in code for BI an AI journal support. | |
| 0000 | 110 | | | | |
| 0000 | 111 | V02-020 | RAS0063 | Ron Schaefer | 29-Jan-1982 |
| 0000 | 112 | | | Correct probes of the user's key and record buffers. | |
| 0000 | 113 | | | | |
| 0000 | 114 | V02-019 | CDS0077 | C. D. Saether | 24-Feb-1981 |
| | | | | | 10:35 |

0000 115 : If really sequential file, specify noread to cache.
0000 116 : Note that this works correctly only for 512 byte fixed
0000 117 : length records, with the "bucket size" at one block.
0000 118 :
0000 119 : V02-018 CDS0076 C. D. Saether 07-Oct-1980 11:05
0000 120 : Release auto-locked record if rm\$getrec2_put did not release
0000 121 : it (because it was current record).
0000 122 :
0000 123 : V02-017 REFORMAT K. E. Kinnear 31-Jul-1980 9:05
0000 124 :
0000 125 : V01-016 CDS0075 C. D. Saether 20-Jan-1980 11:40
0000 126 : Fix bug so manually locked record is released on error.
0000 127 :
0000 128 : V01-015 CDS0042 C. D. Saether 12-Oct-1979 17:40
0000 129 : Update ebk correctly for seq file extend.
0000 130 :
0000 131 : V01-014 JAK0020 J. A. Krycka 11-Sep-1979 10:00
0000 132 : Remove network code.
0000 133 :
0000 134 : V01-013 CDS0024 C. D. Saether 27-Jul-79 5:05
0000 135 : Fudge up code so it works with shared fix length seq files.
0000 136 :
0000 137 : V01-012 WSK0001 W. S. Koenig 22-Dec-1978 11:20
0000 138 : Fixed bug destroying r4 when extend failed.
0000 139 :
0000 140 : V01-011 RAN0003 R. A. Newell 9-Nov-1978 10:56
0000 141 : File sharing code enhancements.
0000 142 :--
0000 143 :--

0000 145 .SBTTL DECLARATIONS
0000 146
0000 147 :
0000 148 : Include Files:
0000 149 :
0000 150 :
0000 151 :
0000 152 : Macros:
0000 153 :
0000 154 :
0000 155 \$IFBDEF
0000 156 \$SBDBDEF
0000 157 \$CSHDEF
0000 158 \$DLCDDEF
0000 159 \$FABDEF
0000 160 \$FIBDEF
0000 161 \$RABDEF
0000 162 \$IRBDEF
0000 163 \$RMSDEF
0000 164 \$RJRDEF
0000 165 \$CJFDEF
0000 166 :
0000 167 : Equated Symbols:
0000 168 :
0000 169 :
0000 170 :
00000020 0000 171 ROP=RAB\$L_ROP*8 ; bit offset to rop field
0000 172 :
0000 173 :
0000 174 : Own Storage:
0000 175 :
0000 176 :

0000 178 .SBTTL RM\$PUT2 - HIGH LEVEL RELATIVE \$PUT
0000 179
0000 180 :++
0000 181 : RM\$PUT2 -- High Level Relative \$PUT.
0000 182 :
0000 183 : This module performs the following functions:
0000 184 :
0000 185 : 1. Calls RMSGETREC2_PUT to gain access to the bucket, locking it,
0000 186 : and unlocking any record automatically locked. The unlocking
0000 187 : is deferred if key access and the record is the current record
0000 188 : to avoid opening a window where the record is unlocked while the
0000 189 : bucket is being reaccessed.
0000 190 :
0000 191 : 2. If the return from RMSGETREC2_PUT indicates that the desired
0000 192 : record is past the current end of file, calls RMSEXTEND2 to extend
0000 193 : the file and tries again
0000 194 :
0000 195 : 3. If manual locking is specified, the record to be written is locked,
0000 196 : otherwise the routine merely checks that the record is not locked
0000 197 : by another stream. If the record was not unlocked in rm\$getrec2_put
0000 198 : (unlock_rp still set), it is unlocked at this point.
0000 199 :
0000 200 : 4. The record is checked for non-existence and if so the record is
0000 201 : copied to the bucket buffer.
0000 202 :
0000 203 : 5. Access to the bucket is released, causing the buffer to be written
0000 204 : unless deferred write has been specified (at open time).
0000 205 :
0000 206 : Calling Sequence:
0000 207 :
0000 208 : Entered via case branch from RM\$PUT at RM\$PUT2.
0000 209 :
0000 210 : Input Parameters:
0000 211 :
0000 212 : R11 impure area address
0000 213 : R10 ifab addr
0000 214 : R9 irab addr
0000 215 : R8 rab addr
0000 216 :
0000 217 : Implicit Inputs:
0000 218 :
0000 219 : The contents of the rab and related irab and ifab.
0000 220 :
0000 221 : Output Parameters:
0000 222 :
0000 223 : R1 thru R7 destroyed
0000 224 : R0 status
0000 225 :
0000 226 : Implicit Outputs:
0000 227 :
0000 228 : Various fields of the rab are filled in to reflect
0000 229 : the status of the operation (see functional spec
0000 230 : for details).
0000 231 :
0000 232 : The irab is similarly updated.
0000 233 :
0000 234 : Completion Codes:

PSE
---RMS
SAEPha
---Ini
ComPas
SymPas
SymPse
CraAss
The797
The

682

27

Mac
----\$2
-\$2

TOT

16C

The

MAC

0000 235 :
0000 236 : Standard rms (see functional spec).
0000 237 :
0000 238 : Side Effects:
0000 239 :
0000 240 : none
0000 241 :
0000 242 :--
0000 243 :

03 6A 38 0000 245 RM\$PUT2:
 53 04 E1 0000 246 \$TSTPT PUT2
 FFED' 30 0009 247 \$CSHFLAGS LOCK
 03 50 E8 0000 248 #IFBSV_SEQFIL,(R10),10\$; require lock on bucket
 0088 31 0010 249 BISL2 #CSH\$M_NOREAD,R3 ; check if really sequential
 0019 250 10\$: BSBW RM\$GETREC2_PUT ; don't read block if really seq.
 0019 251 CHKERR: BLBS R0,10\$; go access bucket
 0019 252 BRW CHKEOF ; continue if success
 0019 253 : ; otherwise go check if EOF
 0019 254 : Handle record locking, if required.
 0019 255 :
 0019 256 : If automatic locking (RAB\$V_ULK = 0), need merely check that record
 0019 257 : is not locked since bucket is locked (and no other user could possibly
 0019 258 : lock the record until the bucket is released).
 0019 259 :
 0019 260 : If manual unlocking (RAB\$V_ULK = 1), must lock the record.
 0019 261 :
 0019 262 :
 53 6A 33 E0 0019 263 10\$: BBS #IFBSV_NORECLK,(R10),CHKCTL; branch if no locking
 51 48 A9 D0 001D 264 MOVL IRBSL_RP(R9),R1 ; get record #
 52 D4 0021 265 CLRL R2 ; zero hi half
 0023 266 :
 0023 267 : If record was previously auto-locked and unlocking was deferred to avoid
 0023 268 : and unlocked record window, it is unlocked at this point.
 0023 269 :
 0023 270 :
 07 69 2D E5 0023 271 BBCC #IRBSV_UNLOCK_RP,(R9),20\$; branch if already unlocked
 06 BB 0027 272 PUSHR #^M<R1,R2> ; save these
 FFD4' 30 0029 273 BSBW RMSUNLOCK ; unlock the record
 06 BA 002C 274 POPR #^M<R1,R2> ; restore registers
 21 68 32 E0 002E 275 20\$: BBS #RAB\$V_ULK+ROP,(R8),LOCK ; branch if manual locking
 FFCB' 30 0032 276 BSBW RMSQUERY_LCK ; o.k. to write?
 8021 8F 50 81 0035 277 CMPW R0,#RMSS_OK_RLK&^xFFFF ; only read allowed?
 1A 12 003A 278 BNEQ CHKLCK ; branch if not (so far so good)
 003C 279 RM\$ERR RLK ; switch status to error
 0041 280 :
 0041 281 : Handle error.
 0041 282 :
 0041 283 :
 0041 284 :
 54 20 A9 D0 0041 285 CLEAN1: MOVL IRBSL_CURBDB(R9),R4 ; Update R4 in case BDB was released.
 57 50 D0 0045 286 MOVL R0,R7 ; save status code
 04 68 32 E1 0048 287 CLEAN2:
 FFAD' 31 0048 288 BBC #RAB\$V_ULK+ROP,(R8),10\$; this record manually locked?
 004C 289 SSB #IRBSV_UNLOCK_RP,(R9) ; yes, make sure it's released
 0050 290 10\$: BRW RM\$CLN2_PUT ; go clean up
 0053 291 :
 0053 292 :
 0053 293 : Manual locking. Must lock the record.
 0053 294 :
 0053 295 :
 8061 8F FFAA' 30 0053 296 LOCK: BSBW RMSLOCK ; go lock record
 E8 50 E9 0056 297 CHKLCK: BLBC R0,CLEAN1 ; branch on failure
 50 B1 0059 298 CMPW R0,#RMSS_OK_WAT&^xFFFF ; did we wait for it?
 10 12 005E 299 BNEQ CHKCTL ; branch if not
 0060 300 :
 0060 301 :

0060 302 : Lock routine stalled, this means that the bucket was
0060 303 : released, go re-access bucket.
0060 304 :
0060 305 :
03 6A 38 E1 0063 306 \$CSHFLAGS LOCK
53 04 C8 0067 307 BBC #IFB\$V_SEQFIL,(R10),10\$: require lock on bucket
FF93. 30 006A 308 BISL2 #CSH\$M_NOREAD,R3 : check if really sequential
D1 50 E9 006D 309 10\$: BSBW RMS\$GETREC2_PUF : don't read block if really seq.
0070 310 BLBC R0,CLEAN1 : go re-access bucket
0070 311 : get out on error
0070 312 :
0070 313 : Locking all set.
0070 314 : Check for record already existent and if not, copy the record to the buffer.
0070 315 :
0070 316 :
0070 317 CHKCTL:
09 6A 38 E0 0070 318 BBS #IFB\$V_SEQFIL,(R10),10\$: if seq file no control byte
08 65 91 0074 319 CMPB (R5),#BLCSM_REC : does record exist?
04 12 0077 320 BNEQ 10\$: branch if not (ok to put)
1D 68 24 E1 0079 321 BBC #RAB\$V_UIF+ROP,(R8),ERRREX: error unless uif bit set
5C 01 D0 007D 322 10\$: MOVL #1_AP : indicate to PUTUPD2 this is a PUT
0082 30 0080 323 BSBW RMS\$PUTUPD2 : go copy record
11 57 E9 0083 324 BLBC R7,30\$: branch on error
0086 325 :
0086 326 ASSUME RAB\$C_SEQ EQ 0
0086 327 :
1E A8 95 0086 328 TSTB RAB\$B_RAC(R8) : sequential access?
06 12 0089 329 BNEQ 20\$: branch if not
40 A9 01 48 A9 C1 0088 330 ADDL3 IRBSL_RP(R9),#1,IRBSL_NRP(R9); yes - set nrp from rp + 1
48 A9 D4 0091 331 20\$: CLRL IRBSL_RP(R9) : show no current record
FF69. 31 0094 332 BRW RMS\$RL52 : go finish up
FF66. 31 0097 333 30\$: BRW RMS\$CLN2_PUT : clean up on error
009A 334 :
009A 335 :
009A 336 : Record already exists. Declare error and go clean up.
009A 337 :
009A 338 :
009A 339 ERRREX:
A7 11 009A 340 RMSERR REX,R7 : set error code
009F 341 BRB CLEAN2 : go clean up
00A1 342 :
00A1 343 :
00A1 344 : Check if error from RMS\$GETREC2_PUT is due to eof, and if so extend the file.
00A1 345 :
00A1 346 :
00A1 347 CHKEOF:
827A 8F 50 B1 00A1 348 CMPW R0,#RMSS_EOF&^xFFFF : is error = eof?
7A 12 00A6 349 BNEQ 10\$: branch if not
56 52. D0 00A8 350 MOVL R2,R6 : save desired hi vbn + 1
FF52. 30 00AB 351 BSBW RMS\$LOCK_PROLOG : lock vbn 1
71 50 E9 00AE 352 BLBC R0,10\$: branch on error
6C AA 54 D0 00B1 353 MOVL R4,IFBSL_LOCK_BDB(R10) : save bdb addr
00B5 354 :
00B5 355 :
00B5 356 : Prolog is now interlocked, thus preventing other extends.
00B5 357 : Check that extend is still required.
00B5 358 :

56 74 AA D1 00B5 359
 6A 1E 00B9 360 CMPL IFBSL_EBK(R10),R6 ; still need to extend eof?
 56 D7 00BB 361 BGEQU 20\$ branch if not
 56 70 AA C2 00BD 362 DECL R6 adjust for hbk
 67 1B 00C1 363 SUBL2 IFBSL_HBK(R10),R6 compute # of blocks needed
 00C3 364 BLEQU 25\$ branch if none (need only format)
 00C3 365
 00C3 366 :
 00C3 367 : Allocate a fib to do the extend.
 00C3 368 :
 00C3 369
 52 40 54 DD 00C3 370 PUSHL R4 ; save lock bdb addr around calls
 FF34' 30 00C9 371 MOVZBL #FIB\$C_LENGTH,R2 ; size of fib
 4C 50 E9 00CC 372 BSBW RMSGET5PC1 ; go allocate fib
 18 A1 4C AA 3C 00CF 373 BLBC R0,8\$ branch on failure
 04 12 00D4 374 MOVZWL IFBSW_RTDEQ(R10),FIBSL_EXSZ(R1); set default extend size
 16 A1 08 88 00D6 375 BNEQ 2\$ branch if non-zero
 56 18 A1 D1 00DA 376 BISB2 #FIB\$M_ALDEF,FIB\$W_EXCTL(R1); flag maximize with vol. default
 04 1E 00DE 377 2\$: CMPL FIBSL_EXSZ(R1),R6 ; is default > # blocks needed?
 18 A1 56 D0 00E0 378 BGEQU 4\$; branch if yes
 00E4 379 MOVL R6,FIBSL_EXSZ(R1) ; no - use required extend size
 00E4 380 4\$:
 00E4 381
 00E4 382 : Do the extend.
 00E4 383 :
 00E4 384 :
 00E4 385
 20 A9 D4 00E4 386 CLRL IRBSL_CURBDB(R9) ; zero current bdb
 FF16' 30 00E7 387 BSBW RMSEXTEND0_ALT ; do extend and deallocate fib
 2E 50 E9 00EA 388 BLBC R0,8\$; branch on error
 00ED 389
 00ED 390 : If extend worked we can get rid of bdb addr (r4) on top of stack.
 00ED 391 :
 00ED 392 :
 00ED 393 :
 8E D5 00ED 394 TSTL (SP)+
 00EF 395
 00EF 396 :
 00EF 397 : Format the buckets (i.e., write zeroes) and update prolog.
 00EF 398 :
 00EF 399 :
 00EF 400 15\$:
 16 6A 38 E0 00EF 401 BBS #IFBSV_SEQFIL,(R10),7\$; if seq don't zero
 FF0A' 30 00F3 402 BSBW RMSFMT_BKT2 write zeroed blocks
 FF07' 30 00F6 403 BSBW RMSUPD_PROLOG2 update prolog
 26 50 E9 00F9 404 BLBC R0,10\$ branch on error
 00FC 405 3\$: \$CSHFLAGS LOCK specify lock required
 51 44 A9 D0 00FF 406 5\$: MOVL IRBSL_CURVBN(R9),R1 set vbn
 FEFA' 30 0103 407 BSBW RMSREADBKT2 go access bucket
 FFOA 31 0106 408 BRW CHKERR and try again
 54 6C AA D0 ^109 409 7\$: MOVL IFBSL_LOCK_BDB(R10),R4 ; restore r4
 FEF0' 30 010D 410 BSBW RMSSETHEBK update eof
 74 AA 01 44 A9 C1 0110 411 ADDL3 IRBSL_CURVBN(R9),#1,IFBSL_EBK(R10); just make ebk beyond block
 0116 412
 0116 413
 0116 414
 0116 415 \$CSHFLAGS <LOCK,NOREAD> ; no need to read block if seq file.

E4 11 0119 416 BRB 5\$: try again
011B 417
011B 418 ;
011B 419 ; Error has occurred allocating a fib or extending the file.
011B 420 ;
011B 421
011B 422 8\$:
10, BA 011B 423 POPR #^M<R4> ; restore bdb addr for cleanup
FEE0, 30 011D 424 BSBW RM\$RLSPLG ; release lock on prolog
54 D4 0120 425 CLRL R4 ; don't release bdb twice
FF1C 31 0122 426 10\$: BRW CLEAN1 ; clean up on error
0125 427
0125 428 ;
0125 429 ; Another user has done an extend since we last checked the eof data
0125 430 ; (hard to believe he could sneak in that window, isn't it?)
0125 431
0125 432 ; Therefore, our work is already done. We need merely unlock the prolog
0125 433 ; and go try our put again.
0125 434 ;
0125 435
FED8, 30 0125 436 20\$: BSBW RM\$RLSPLG ; unlock the prolog
D2 11 0128 437 BRB 3\$; continue with put
012A 438
012A 439 ;
012A 440 ; File needs no extending, only formatting.
012A 441 ;
012A 442
56 70 AA 01 C1 012A 443 25\$: ADDL3 #1,IFBSL_HBK(R10),R6 ; set end vbn of extent + 1
51 74 AA D0 012F 444 MOVL IFBSL_EBR(R10),R1 ; and start vbn of extent
BA 11 0133 445 BRB 15\$; go format buckets
0135 446

0135 448 .SBTTL RM\$PUTUPD2 - COMMON \$PUT AND \$UPDATE RELATIVE ROUTINE
0135 449
0135 450 ++
0135 451 RM\$PUTUPD2 -- Common \$PUT and \$UPDATE Relative Routine.
0135 452
0135 453 This routine:
0135 454
0135 455 1. Saves r0 status code in r7.
0135 456
0135 457 2. Verifies the rsz and rbf parameters, as well as rhb if rfm=vfc.
0135 458
0135 459 3. Set the delete control byte to say record exists.
0135 460
0135 461 4. Store record size unless rfm=fix.
0135 462
0135 463 5. If rfm=vfc, copy the rhb to buffer.
0135 464
0135 465 6. Copy the record to the buffer and set the valid and dirty
0135 466 buffer flags.
0135 467
0135 468 Calling Sequence:
0135 469
0135 470 BSBW RM\$PUTUPD2
0135 471
0135 472 Input Parameters:
0135 473
0135 474 AP non 0 if called from \$put, 0 if from \$update
0135 475 R8-R11 same as for entry at rm\$put2
0135 476 R5 address of record in bucket buffer
0135 477 R4 bdb address
0135 478 R0 status code
0135 479
0135 480 Implicit Inputs:
0135 481
0135 482 The contents of the various control blocks, in particular:
0135 483
0135 484 RAB\$W_RSZ record size
0135 485 RAB\$L_RBF record address
0135 486 RAB\$L_RHB record header buffer address if rfm=vfc
0135 487 IFB\$B_RFM record format
0135 488 IFB\$W_MRS maximum record length
0135 489 IFB\$B_FSZ fixed header size if rfm=vfc
0135 490
0135 491 Output Parameters:
0135 492
0135 493 R7 status code
0135 494 R0-R3,R5,R6 destroyed
0135 495
0135 496 Implicit Outputs:
0135 497
0135 498 none.
0135 499
0135 500 Completion Codes:
0135 501
0135 502 Standard rms, in particular the code from r0 on input or rsz, rbf,
0135 503 or rhb.
0135 504

0135 505 ; Side Effects:
0135 506 ;
0135 507 ; none.
0135 508 ;--
0135 509

06 00A0 57 50 0135 511 RMSPUTUPD2::
 42 00A2 CA 02 0135 512 MOVL R0, R7 : copy status code
 E0 0138 513 BBS #IFBSV_BI, IFBSB_JNLFLG(R10), 5\$: BI journaling on?
 E1 013E 514 BBC #IFBSV_RUP, IFBSB_JNLFLG2(R10), 40\$; RUP? If no BI/RUP, skip this
 30 BB 0144 515 SS: PUSHR #^M<R4, R5>
 0124 30 0146 516 BSBW MAKEJNL ; save R4, R5
 5C D5 0149 517 TSTL ; set up journal record
 06 13 014B 518 BEQL 10\$; doing \$PUT?
 14 A6 0048 8F B0 014D 519 MOVW #RJRSC_RECLEN, BDBSW_NUMB(R6) ; branch if update
 54 6E D0 0153 520 10\$: MOVL (SP), R4 ; don't write empty cell
 11 00A0 CA 02 E1 0156 521 BBC #IFBSV_BI, IFBSB_JNLFLG(R10), 20\$; restore R4
 56 DD 015C 522 PUSHL R6 ; branch if no BI
 7E 02 9A 015E 523 MOVZBL #CJFS_BI, -(SP) ; jBDB arg
 00000000'EF 16 0161 524 JSB RMSWRTJNL ; specify BI
 5E 08 C0 0167 525 ADDL2 #8, SP ; write journal record
 14 50 E9 016A 526 BLBC R0, 30\$; discard arglist
 0E 00A2 CA 02 E1 016D 527 20\$: BBC #IFBSV_RUP, IFBSB_JNLFLG2(R10), 30\$; get out on error
 56 DD 0173 528 PUSHL R6 ; branch if not RUP
 7E 01 9A 0175 529 MOVZBL #CJFS_RU, -(SP) ; jBDB arg
 00000000'EF 16 0178 530 JSB RMSWRTJNL ; specify BI
 5E 08 C0 017E 531 ADDL2 #8, SP ; write journal record
 30 BA 0181 532 30\$: POPR #^M<R4, R5> ; discard arglist
 6D 50 E9 0183 533 BLBC R0, ERRJNL ; get out on error
 0186 534
 56 22 A8 3C 0186 535 40\$: MGvZWL RABSW_RSZ(R8), R6 ; get record size
 01 50 AA 91 018A 536 CMPB IFBSB_RFMOORG(R10), #FABSC_FIX ; rfm = fix?
 06 13 018E 537 BEQL 50\$; branch of yes
 56 60 AA B1 0190 538 CMPW IFBSW_MRS(R10), R6 ; record too long?
 06 1E 0194 539 BGEQU 60\$; branch if ok
 0196 540
 56 60 AA B1 0196 541 50\$: CMPW IFBSW_MRS(R10), R6 ; 'else fall thru - will be checked
 4B 12 019A 542 BNEQ ERRRSZ ; rsz = fixed record length?
 03 6A 38 E1 019C 543 60\$: BBC #IFBSV_SEQFIL, (R10), CHKVFC ; branch if not
 0082 31 01A0 544 BRW SAVR45 ; let's move record and be done
 03 50 AA 91 01A3 545 CHKVFC: CMPB IFBSB_RFMOORG(R10), #FABSC_VFC ; rfm = vfc?
 70 12 01A7 546 BNEQ SETCTC ; branch if not
 01A9 547
 01A9 548 :
 01A9 549 : Record format is vfc.
 01A9 550 :
 01A9 551 : Probe the record header buffer and copy to bucket.
 01A9 552 :
 01A9 553 :
 51 5F AA 9A 01A9 554 MOVZBL IFBSB_FSZ(R10), R1 ; get fixed header size
 50 2C A8 D0 01AD 555 MOVL RABSL_RHB(R8), R0 ; get the rhb address
 07 13 01B1 556 BEQL 10\$; branch if none
 85 85 08 90 01BA 557 IFNORD R1, (R0), ERRRRHB, IRBSB_MODE(R9); branch if not readable
 51 56 A1 01BD 558 10\$: MOVBL #DLC\$M REC, (R5)+ ; say record exists
 30 BB 01C1 559 ADDW3 R6, R1, (R5)+ ; set rec length = fixed + var
 50 D5 01C3 560 PUSHR #^M<R4, R5> ; save R4, R5
 06 13 01C5 561 TSTL R0 ; rhb speced?
 65 60 51 28 01C7 562 BEQL 20\$; branch if not
 0F 11 01CB 563 MOVC3 R1, (R0), (R5) ; copy rhb
 01CD 564 BRB 40\$
 01CD 565
 01CD 566 :
 01CD 567 : Rhb = 0. Zero the header if doing \$PUT, skip it if \$UPDATE.

01CD 568 ;
 01CD 569 ;
 50 05 05 12 01CD 570 20\$: TSTL AP ; doing \$put?
 51 51 00 11 01D1 571 BNEQ 30\$; branch if yes
 09 09 00 11 01D4 572 ADDL2 R1 R5 ; skip over header
 65 51 00 6E 00 2C 01D6 573 BRB 50\$;
 55 55 53 00 01DC 574 30\$: MOVC5 #0,(SP),#0,R1,(R5) ; zero the header
 46 11 01DF 575 40\$: MOVL R3,R5 ; update buffer address
 01E1 576 50\$: BRB MOVREC ; go move record
 01E1 577 ;
 01E1 578 ;
 01E1 579 ; Handle errors.
 01E1 580 ;
 01E1 581 ;
 01E1 582 ;
 01E1 583 ERRRB: ;
 01E1 584 RMSERR RHB,R7 ; bad record header buffer
 05 01E6 585 RSB ;
 01E7 586 ;
 01E7 587 ERRRSZ: ;
 01E7 588 RMSERR RSZ,R7 ; invalid record length
 05 01EC 589 RSB ;
 01ED 590 ;
 01ED 591 ERRRB: ;
 01ED 592 RMSERR RBF,R7 ; invalid record header buffer
 05 01F2 593 RSB ;
 01F3 594 ;
 57 50 00 01F3 595 ERRJNL: ;
 01F3 596 MOVL R0,R7 ;
 05 01F6 597 RSB ;
 01F7 598 ;
 01F7 599 ;
 01F7 600 ; Probe readability of all pages (> 2) of user record.
 01F7 601 ;
 01F7 602 ;
 01F7 603 LONG_PROBE: ;
 50 50 56 00 01F7 604 MOVL R6,R0 ; copy buffer length
 51 51 53 00 01FA 605 MOVL R3,R1 ; and address
 52 FE00 8F 32 01FD 606 CVTWL #-512,R2 ; set address constant
 0202 607 10\$: IFNORD R0,(R1),ERRRB,IRB\$B_MODE(R9); branch if not readable
 51 51 52 C2 0209 608 SUBL2 R2,R1 ; get address next page
 50 6042 3E 020C 609 MOVAW (R0)[R2],R0 ; adjust remaining length
 F0 14 0210 610 BGTR 10\$; loop if more to do
 50 52 C2 0212 611 SUBL2 R2,R0 ; need to handle last page?
 EB 14 0215 612 BGTR 10\$; branch if yes
 24 11 0217 613 BRB MOVREC1 ; rejoin main sequence
 0219 614

| | | |
|------------------|-----------|--|
| 02 85 08 90 | 0219 616 | |
| 02 50 AA 91 | 0219 617 | |
| 03 12 0220 | 0219 618 | : Set "record exists" into control byte, store the record size if var record |
| 85 56 B0 0222 | 0219 619 | : format, and move the record. |
| 53 28 B8 DE 0227 | 0219 620 | : |
| 56 B5 022B | 0219 621 | |
| 0E 13 022D | 0219 622 | SETCTL: MOVB #DLC\$M REC, (R5) + |
| 56 B1 022F | 021C 623 | CMPB IFBSB RFMORG(R10), #FAB\$C ; say record exists |
| C1 1A 0234 | 0220 624 | BNEQ SAVR45 ; branch if not |
| 0200 8F 56 | 0225 625 | MOVW R6, (R5) + |
| 56 022B | 0227 626 | SAVR45: PUSHR #^M<R4, R5> ; store record length |
| 0200 8F 56 | 022D 627 | MOVREC: MOVAL @RAB\$L_RBF(R8), R3 ; save R4, R5 |
| C1 1A 0234 | 022F 628 | TSTW R6 ; get buffer addr |
| 0200 8F 56 | 0234 629 | BEQL R6, #512 ; rsz = 0? |
| C1 1A 0236 | 0236 630 | CMPW R6, #512 ; record > 2 pages in length? |
| 0200 8F 56 | 023D 631 | BGTRU LONG PROBE ; branch if yes |
| C1 1A 023D | 023D 632 | IFNORD R6, (R3), ERRRBF, IRBSB_MODE(R9); branch if not readable |
| 65 63 56 28 | 023D 633 | MOVREC1: MOV C3 R6, (R3), (R5) ; move the record |
| 54 6E D0 0241 | 0241 634 | MOVL (SP), R4 ; restore R4 |
| 0A A4 03 88 | 0244 635 | BISB #BDB\$M VAL!BDBSM_DRT, BDB\$B FLGS(R4); say valid and dirty |
| 55 1C 00A0 CA 03 | 0244 636 | #IFBSV-AI, IFBSB JNLFLG(R10), 10\$; branch if not AI journaling |
| 18 A4 4C A9 | 0248 637 | BBC ADDL3 IRBSL RP_OFF(R9), BDBSL_ADDR(R4), R5 ; get cell address |
| 17 10 0254 | 024E 638 | BSBB MAKEJNL ; set up journal record |
| 54 6E D0 0256 | 0254 639 | MOVL (SP), R4 ; restore R4 |
| 56 DD 0259 | 0256 640 | PUSHL R6 ; jBDB arg |
| 7E 03 9A 025B | 0259 641 | MOVZBL #CJFS AI, -(SP) ; specify BI |
| 00000000'EF | 16 025E | JSB RMSWRTJNL ; write journal record |
| 5E 08 C0 0264 | 025E 643 | ADDL2 #8, SP ; discard arglist |
| 57 50 D0 0267 | 0264 644 | MOVL R0, R7 ; set status code |
| 30 BA 026A | 0267 645 | POPR #^M<R4, R5> ; restore R4, R5 |
| 05 026C | 026A 646 | RSB |
| | 10\$: 647 | |
| | 20\$: 647 | |

026D 649 :++
 026D 650 : Subroutine to construct journal entry
 026D 651 :
 026D 652 :
 026D 653 : Input: R5 - Cell to journal
 026D 654 :
 026D 655 : Output: R6 - Addr of journaling BDB
 026D 656 : Destroys R1,R2,R4-R5
 026D 657 :
 026D 658 :
 026D 659 :--
 026D 660 :
 026D 661 :
 026D 662 MAKEJNL:
 56 30 A9 D0 026D 663 MOVL IRB\$L_JNLBDB(R9),R6 ; get journaling BDB
 52 18 A6 D0 0271 664 MOVL BDB\$L_ADDR(R6),R2 ; get journaling buffer
 40 A2 48 A9 D0 0275 665 MOVL IRB\$L_RP(R9),RJR\$L_RRN(R2) ; fill in relative record num
 03 A2 02 90 027A 666 MOVB #RJR\$C_RECORD,RJR\$B_ENTRY_TYPE(R2) ; RJR type
 027E 667 :
 027E 668 ASSUME RJR\$B_OPER EQ RJR\$B_ORG+1
 027E 669 :
 1301 8F B0 027E 670 MOVW #<RJR\$ PUT@8 + RJR\$C_REL>,-
 04 A2 0282 671 RJR\$B_ORG(R2) ; fill in file type & oper
 5C D5 0284 672 TSTL AP ; doing \$PUT?
 04 12 0286 673 BNEQ 10\$; branch if so
 05 A2 1C 90 0288 674 MOVW #RJR\$ UPDATE,RJR\$B_OPER(R2) ; indicate \$UPDATE
 46 A2 62 A9 B0 028C 675 10\$: ADDW3 IRB\$W_CSIZ(R9),RJR\$W_RSIZE(R2) ; set cell size
 0048 8F 62 A9 A1 0291 676 IRB\$W_CSIZ(R9),#RJR\$C_RECLEN,-
 14 A6 0297 677 BDB\$W_NUMB(R6) ; set journal record size
 65 62 A9 28 0299 678 MOVC3 IRB\$W_CSIZ(R9), (R5),-
 48 A2 029D 679 RJR\$T_RIMAGE(R2) ; copy entire cell
 05 029F 680 RSB
 02A0 681 :
 02A0 682 .END

| | | | |
|-----------------|---------------|-------------------|----------------|
| \$\$PSECT_EP | = 00000000 | IRBSL_RP_OFF | = 00000040 |
| \$\$TMP | = 00000005 | IRBSV_UNLOCK_RP | = 00000020 |
| \$\$RMSTEST | = 0000001A | IRBSW_CSIZ | = 00000062 |
| \$\$RMS_PBUGCHK | = 00000010 | LOCK | 00000053 R 01 |
| \$\$RMS_TBUGCHK | = 00000008 | LONG_PROBE | 000001F7 R 01 |
| \$\$RMS_UMODE | = 00000004 | MAKEJNL | 00000260 R 01 |
| BDBSB_FLGS | = 0000000A | MOVREC | 00000227 R 01 |
| BDBSL_ADDR | = 00000018 | MOVREC1 | 00000230 R 01 |
| BDBSM_DRT | = 00000002 | PIOSA_TRACE | ***** X 01 |
| BDBSM_VAL | = 00000001 | RABSB_RAC | = 0000001E |
| BDBSW_NUMB | = 00000014 | RABSC_SEQ | = 00000000 |
| CHKCTC | 00000070 R 01 | RABSL_RBF | = 00000028 |
| CHKEOF | 000000A1 R 01 | RABSL_RHB | = 0000002C |
| CHKERR | 00000013 R 01 | RABSL_ROP | = 00000004 |
| CHKLCK | 00000056 R 01 | RABSV_UIF | = 00000004 |
| CHKVFC | 000001A3 R 01 | RABSV_ULK | = 00000012 |
| CJFS_AI | = 00000003 | RABSW_RSZ | = 00000022 |
| CJFS_BI | = 00000002 | RJR\$B_ENTRY_TYPE | = 00000003 |
| CJFS_RU | = 00000001 | RJR\$B_OPER | = 00000005 |
| CLEAN1 | 00000041 R 01 | RJR\$B_ORG | = 00000004 |
| CLEAN2 | 00000048 R 01 | RJR\$C_RECLEN | = 00000048 |
| CSHSM_LOCK | = 00000001 | RJR\$C_RECORD | = 00000002 |
| CSHSM_NOBUFFER | = 00000008 | RJR\$C_REL | = 00000001 |
| CSHSM_NOREAD | = 00000004 | RJR\$L_RRN | = 00000040 |
| DLCSM_REC | = 00000008 | RJR\$T_RIMAGE | = 00000048 |
| ERRJNC | 000001F3 R 01 | RJR\$W_RSIZE | = 00000046 |
| ERRRBF | 000001ED R 01 | RJR\$PUT | = 00000013 |
| ERRREX | 0000009A R 01 | RJR\$UPDATE | = 0000001C |
| ERRRHB | 000001E1 R 01 | RMS\$CN2_PUT | ***** X 01 |
| ERRRSZ | 000001E7 R 01 | RMS\$EXTENDO_ALT | ***** X 01 |
| FABSC_FIX | = 00000001 | RMS\$FMT_BKT2 | ***** X 01 |
| FABSC_VAR | = 00000002 | RMS\$GETREC2_PUT | ***** X 01 |
| FABSC_VFC | = 00000003 | RMS\$GETSPC1 | ***** X 01 |
| FIBSC_LENGTH | = 00000040 | RMS\$LOCK | ***** X 01 |
| FIBSL_EXSZ | = 00000018 | RMS\$LOCK_PROLOG | ***** X 01 |
| FIBSM_ALDEF | = 00000008 | RMS\$PUT2 | 00000000 RG 01 |
| FIBSW_EXCTL | = 00000016 | RMS\$PUTUPD2 | 00000135 RG 01 |
| IFBSB_FSZ | = 0000005F | RMS\$QUERY_LCK | ***** X 01 |
| IFBSB_JNLFLG | = 000000A0 | RMS\$READBKT2 | ***** X 01 |
| IFBSB_JNLFLG2 | = 000000A2 | RMS\$RLS2 | ***** X 01 |
| IFBSB_RFMRG | = 00000050 | RMS\$RLSPLG | ***** X 01 |
| IFBSL_EBK | = 00000074 | RMS\$SETHEBK | ***** X 01 |
| IFBSL_HBK | = 00000070 | RMS\$UNLOCK | ***** X 01 |
| IFBSL_LOCK_BDB | = 0000006C | RMS\$UPD_PROLOG2 | ***** X 01 |
| IFBSV_AI | = 00000003 | RMS\$WRTJNL | ***** X 01 |
| IFBSV_BI | = 00000002 | RMS\$EOF | = 0001827A |
| IFBSV_NORECLK | = 00000033 | RMS\$OK_RLK | = 00018021 |
| IFBSV_RUP | = 00000002 | RMS\$OK_WAT | = 00018061 |
| IFBSV_SEQFIL | = 00000038 | RMS\$RBF | = 00018654 |
| IFBSW_MRS | = 00000060 | RMS\$REX | = 000182A2 |
| IFBSW_RTDEQ | = 0000004C | RMS\$RHB | = 0001866C |
| IRBSB_MODE | = 0000000A | RMS\$RLK | = 000182AA |
| IRBSL_CURBDB | = 00000020 | RMS\$RSZ | = 000186A4 |
| IRBSL_CURVBN | = 00000044 | ROP | = 00000020 |
| IRBSL_JNLBDB | = 00000030 | SAVR45 | 00000225 R 01 |
| IRBSL_NRP | = 00000040 | SETCTL | 00000219 R 01 |
| IRBSL_RP | = 00000048 | TPT\$L_PUT2 | ***** X 01 |

```
+-----+
! Psect synopsis !
+-----+
```

PSECT name

| | Allocation | PSECT No. | Attributes |
|----------|------------------|-----------|---|
| ABS | 00000000 (0.) | 00 (0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| RM\$RMS2 | 000002A0 (672.) | 01 (1.) | PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE |
| SABSS | 00000000 (0.) | 02 (2.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |

```
+-----+
! Performance indicators !
+-----+
```

Phase

| Phase | Page faults | CPU Time | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization | 38 | 00:00:00.08 | 00:00:00.65 |
| Command processing | 112 | 00:00:00.71 | 00:00:04.29 |
| Pass 1 | 385 | 00:00:14.11 | 00:00:44.48 |
| Symbol table sort | 0 | 00:00:02.08 | 00:00:03.97 |
| Pass 2 | 131 | 00:00:02.85 | 00:00:09.51 |
| Symbol table output | 13 | 00:00:00.10 | 00:00:00.13 |
| Psect synopsis output | 2 | 00:00:00.02 | 00:00:00.02 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 683 | 00:00:19.96 | 00:01:03.18 |

The working set limit was 1650 pages.

79795 bytes (156 pages) of virtual memory were used to buffer the intermediate code.

There were 80 pages of symbol table space allocated to hold 1482 non-local and 35 local symbols.

682 source lines were read in Pass 1, producing 15 object records in Pass 2.

27 pages of virtual memory were used to define 26 macros.

```
+-----+
! Macro library statistics !
+-----+
```

Macro library name

| Macro library name | Macros defined |
|-------------------------------------|----------------|
| -\$255\$DUA28:[RMS.OBJ]RMS.MLB;1 | 15 |
| -\$255\$DUA28:[SYS.OBJ]LIB.MLB;1 | 1 |
| -\$255\$DUA28:[SYSLIB]STARLET.MLB;2 | 6 |
| TOTALS (all libraries) | 22 |

1603 GEIS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LI\$:\$:RM2PUT/OBJ=OBJ\$:\$:RM2PUT MSRC\$:\$:RM2PUT/UPDATE=(ENH\$:\$:RM2PUT)+EXECML\$:\$:LIB+LIB\$:\$:RMS/LIB

0323 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

